

FIG. 1

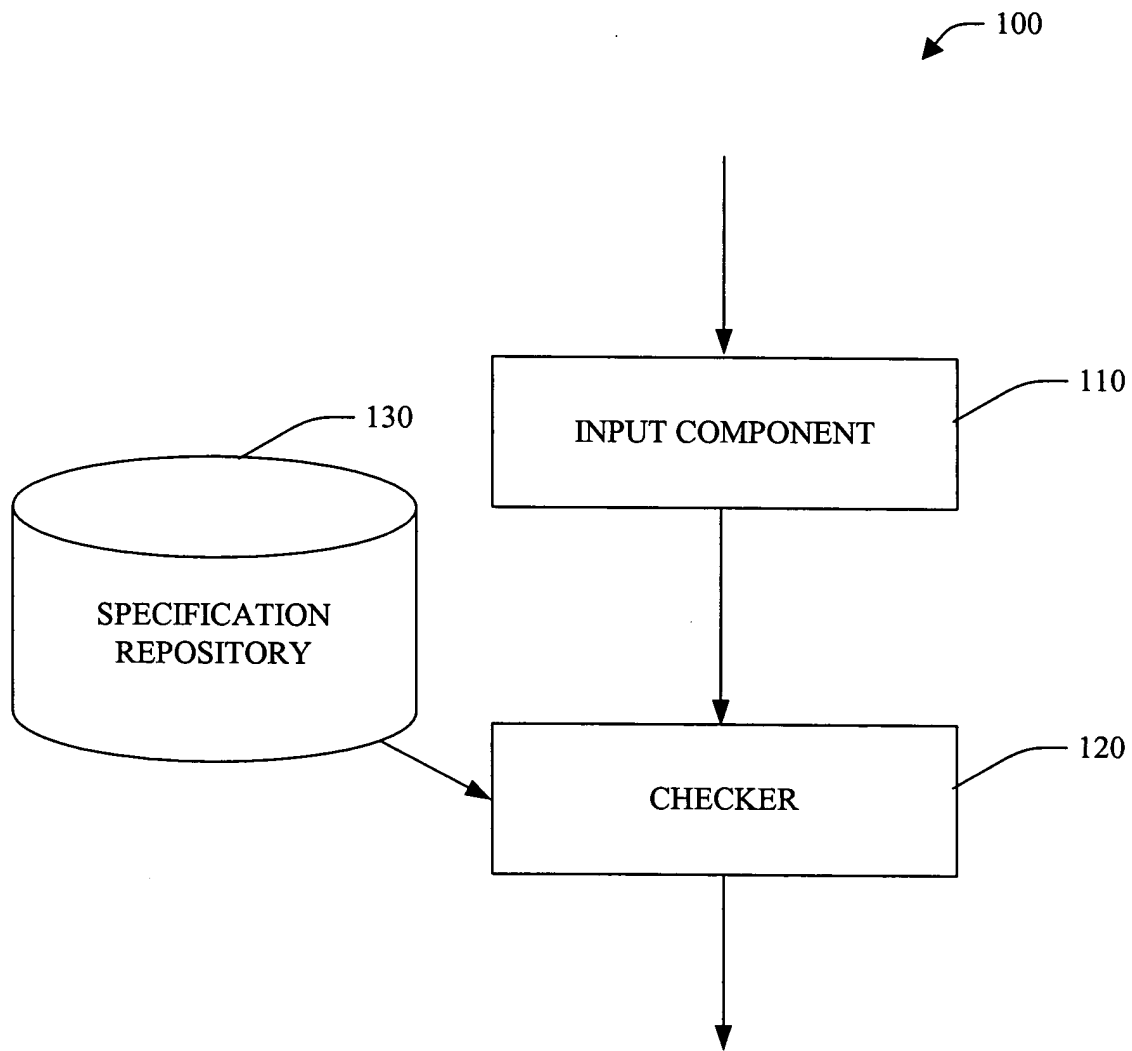


FIG. 2

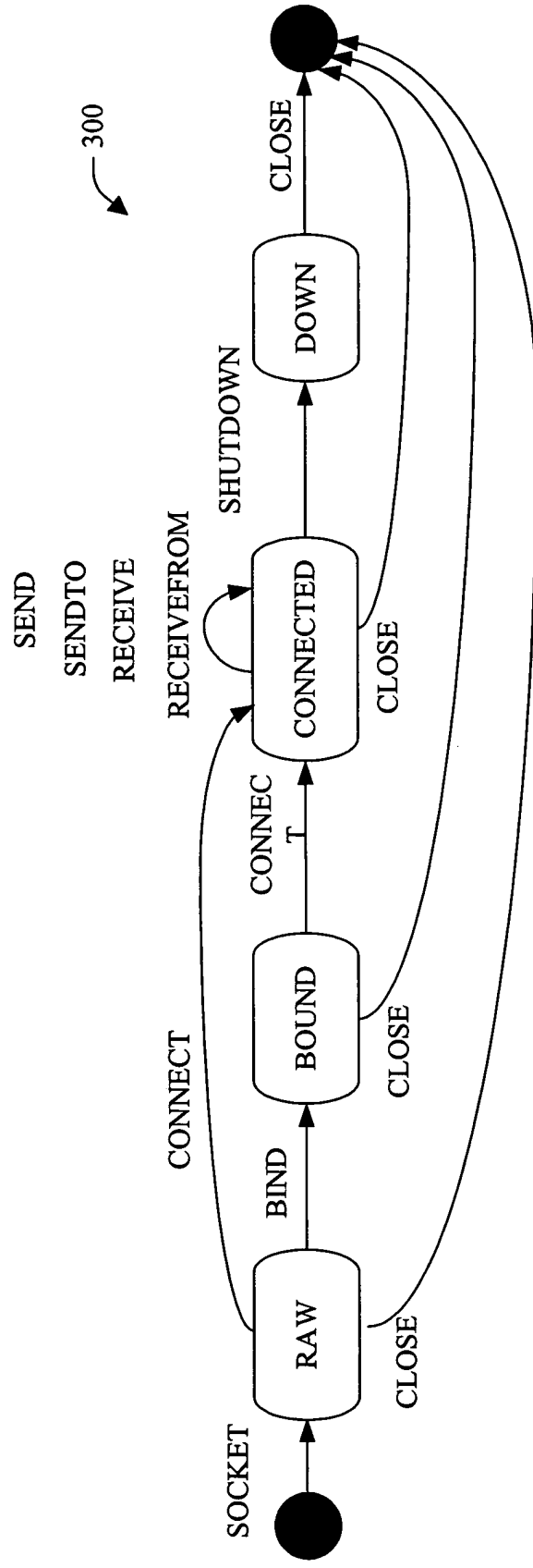


FIG. 3

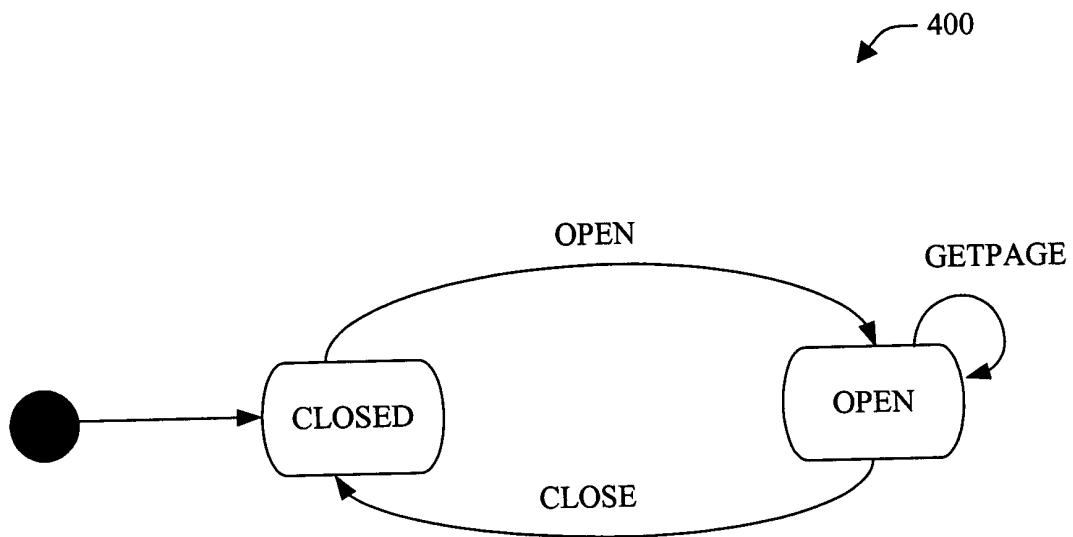


FIG. 4

```
[WithProtocol(  
    CustomState=typeof(SqlConnectionState)) ]  
class SqlConnection  
{  
    [ Creates,  
    OutConnectionState(  
        Status=ConnectionState.Closed,  
        Host="", Database="")]  
    SqlConnection ();  
  
    [ Creates,  
    OutConnectionState(  
        Status=ConnectionState.Closed,  
        StateProvider="NewHostAndDatabase"),  
    OutStateDependsOn("connectionString")]  
    SqlConnection (string connectionString) ;  
  
    [ OutConnectionState(  
        Status=ConnectionState.Open) ]  
    void Open () ;  
}
```

FIG. 5

```

[ WithProtocol(
    CustomStat=typeof(SqlCommandState)) ]
class SqlCommand
{
    [OutCommandState(
        StateProvider="UpdateCommandText"),
        OutStateDependsOn("cmdText") ]
    SqlCommand (string cmdText);

    [ property: Transparent ]
    SqlConnection Connection { get; set; }

    [ InCommandState(
        StateChecker="CheckCommandText"),
        InStateDependsOn("this.Connection") ]
    [ return: OutReaderState(
        StateProvider="GetColumnInfo"),
        OutStateDependsOn("this.Connection","this") ]
    SqlDataReader ExecuteReader ();
}

```

FIG. 6

700

```
{ WithProtocol(  
    CustomState=typeof(sqlReaderState)) ]  
class SqlDataReader  
{  
    [ InReaderState(  
        StateChecker="ValidColumnName"),  
        InStateDependsOn("name") ]  
    object get_Item (string name);  
  
    [ InReaderState(  
        StateChecker="ColumnIsString"),  
        InStateDependsOn("i") ]  
    string GetString (int i);  
}
```

FIG. 7

```

class SqlConnectionState : CustomState
{
    ConnectionState Status
    string Host, Database;

    void NewHostAndDatabase (string connString) {
        // Example plug-in postcondition, which
        // parses a connection string for
        // its host and database names.
        Regex hostRegex = new Regex (
            @"(data source|server)\s*=(\[^\;]*)\b",
            RegexOptions.IgnoreCase);
        Regex dbRegex = new Regex(
            @"(catalog|database)\s*=(\[^\;]*)\b",
            RegexOptions.IgnoreCase);
        for (int i=0; i<connString.Length; i++) {
            MatchCollection dbm =
                hostRegex.Matches(connString[i]);
            if (dbm.Count > 0)
                Host = dbm[0].Groups[2].Captures[0].Value;
            MatchCollection hm =
                dbRegex.Matches(connString[i]);
            if (hm.Count > 0)
                Database = hm[0].Groups[2].Captures[0].Value;
        }
        if (Host == null)
            Fail("could not find host");
        if (Database == null)
            Fail("could not find database");
    }
}

```

FIG. 8

900

```
class SqlCommandState : CustomState
{
    string[] CommandText;

    void UpdateCommandText (string[] c0 { CommandText=c; }

    bool CheckCommandText (SqlConnectionState c) {
        return ISLegalSQL(CommandText, c.Host, c.Database);
    }
}
```

FIG. 9

1000

```
class SqlDataReaderState : CustomState
{
    string [] ColumnNames, ColumnTypes;

    void GetColumnInfor (SqlConnectionState connection,
                        SqlCommandState command) {...}
    bool ValidColumnName (string[] name) {...}
    bool ColumnIsString (int i) {...}
}
```

FIG. 10

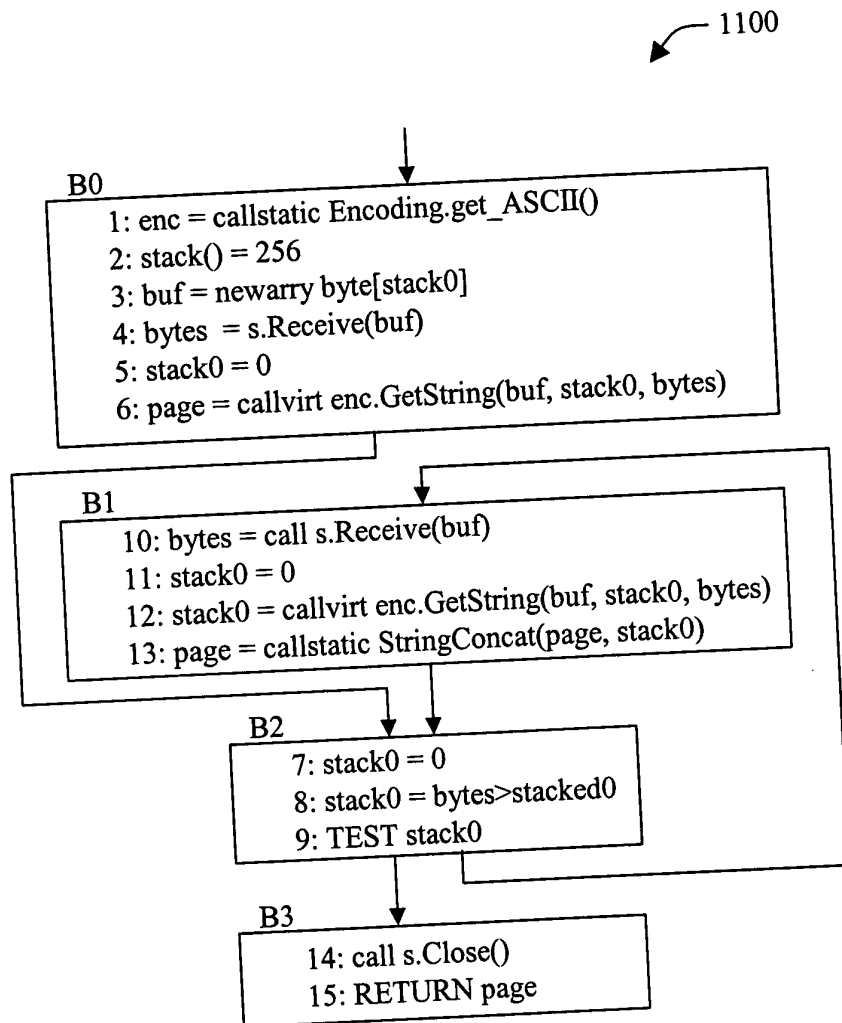


FIG. 11

```

0  s : ref(a0)
1  enc : ref(a1)
2  stack0 : value(int, 256, default)
3  buf : value(byte[], , default)
4  bytes : value(int, , default)
5  stack0 : value(int, 0, default)
6  page : ref(a3)
7  stack0 : value(int, 0, default)
8  stack0 : value(bool, , default)
9  (no change)
10 bytes : value(int, , default)
11 stack0 : value(int, 0, default)
12 stack0 : ref(a4)
13 (no change)
14 (no change)
15 (no change)

```

1200 ↗

```

a0 → (Socket.NotAliased, "connected", 0)
a1 → (Encoding.MaybeAliased/Escaping, default, 0)

a3 → (string.MaybeAliased/Escaping,default, 0)

a4 → (string.MaybeAliased/Escaping,default, 0)
      (a0 removed from capabilities)

```

FIG. 12

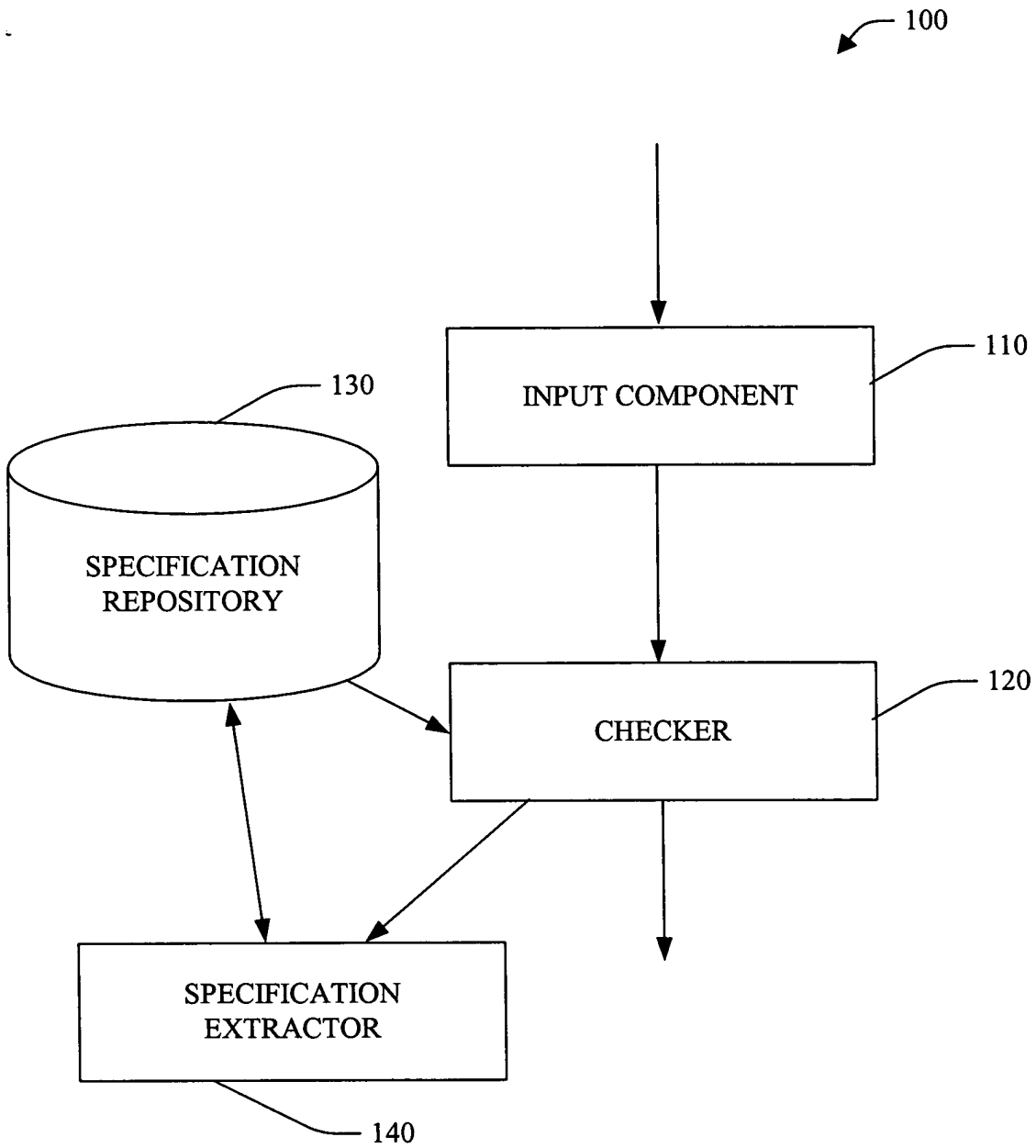


FIG. 13

```
[WithProtocol( UnknownDB, KnownDB)]
class Publications : System.Web.UI.Page
{
    [InConnectionState(WhenEnclosingState=UnknownDB
        Status = ConnectionState.Closed,
        Host = AnyHost, Database = AnyDatabase)
    InConnectionState(WhenEnclosingState=KnownDB
        Status = ConnectionState.Closed,
        Host = XXX, Database = YYY )
    private SqlConnection m_sqlCn;

    [ChangesState( UnknownDB , KnownDB )]
    private void OnPageLoad (EventArgs e)
    {
        m_sqlCn = new SqlConnection(...);
        //...
    }

    [InState( KnownDB )]
    void WriteTRDetail ()
    {
        m_sqlCn.Open();
        SqlCommand objCommand =
            new SqlCommand("EXEC ...", m_sqlCn);
        SqlDataReader objDataReader =
            objCommand.ExecuteReader();
        // ...
    }
}
```

FIG. 14

```

string GetPersonWebURL (
    [ InReaderState(
        ColumnNames = - "internalurl", "externalurl" ",
        ColumnTypes = - "nchar", "nchar" " ]
    SqlDataReader dr )
{
    if (dr["internalurl"] == null)
        if (dr["externalurl"] == null)
            return "";
        else
            // ...
}

```

FIG. 15

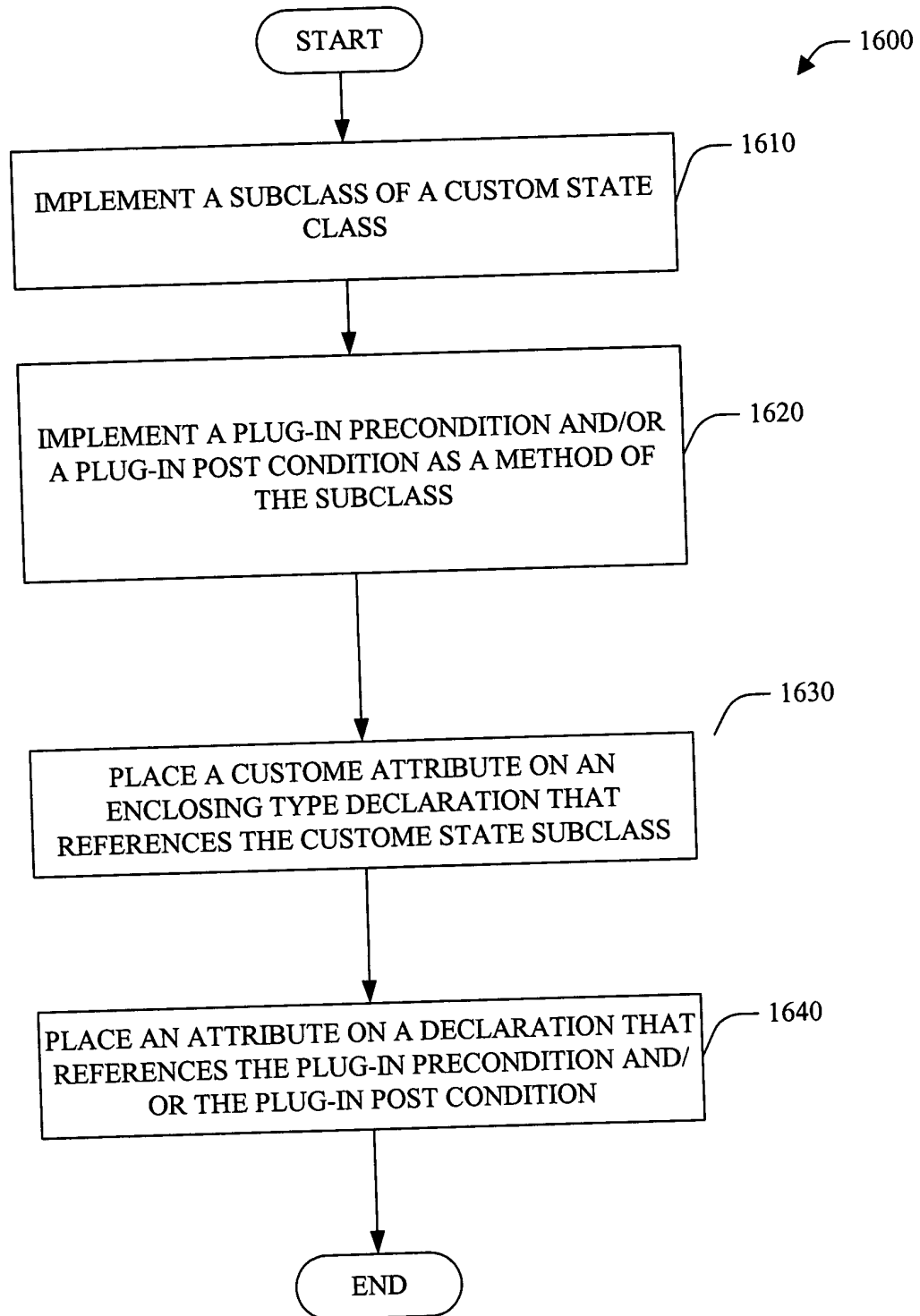


FIG. 16

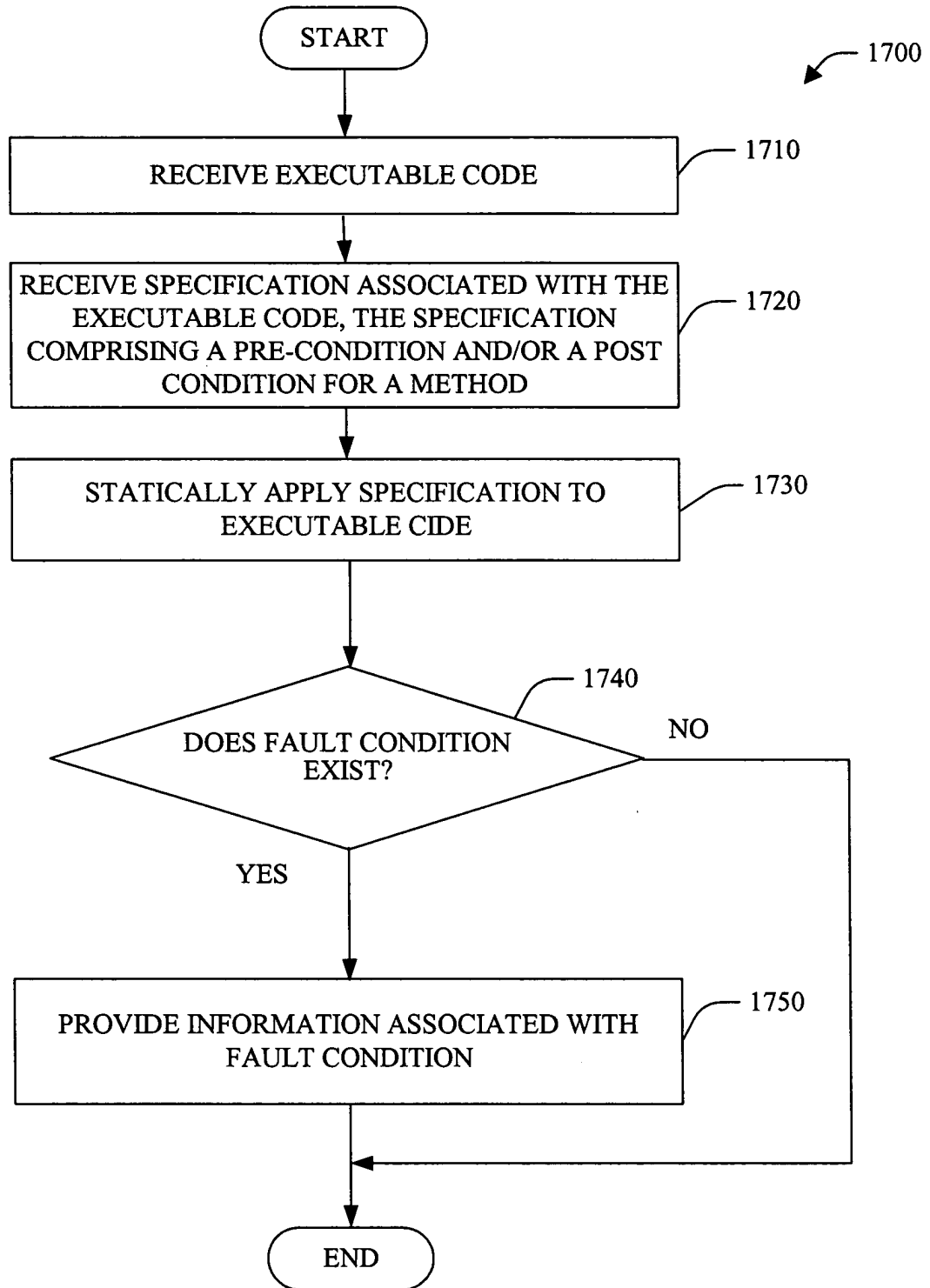


FIG. 17

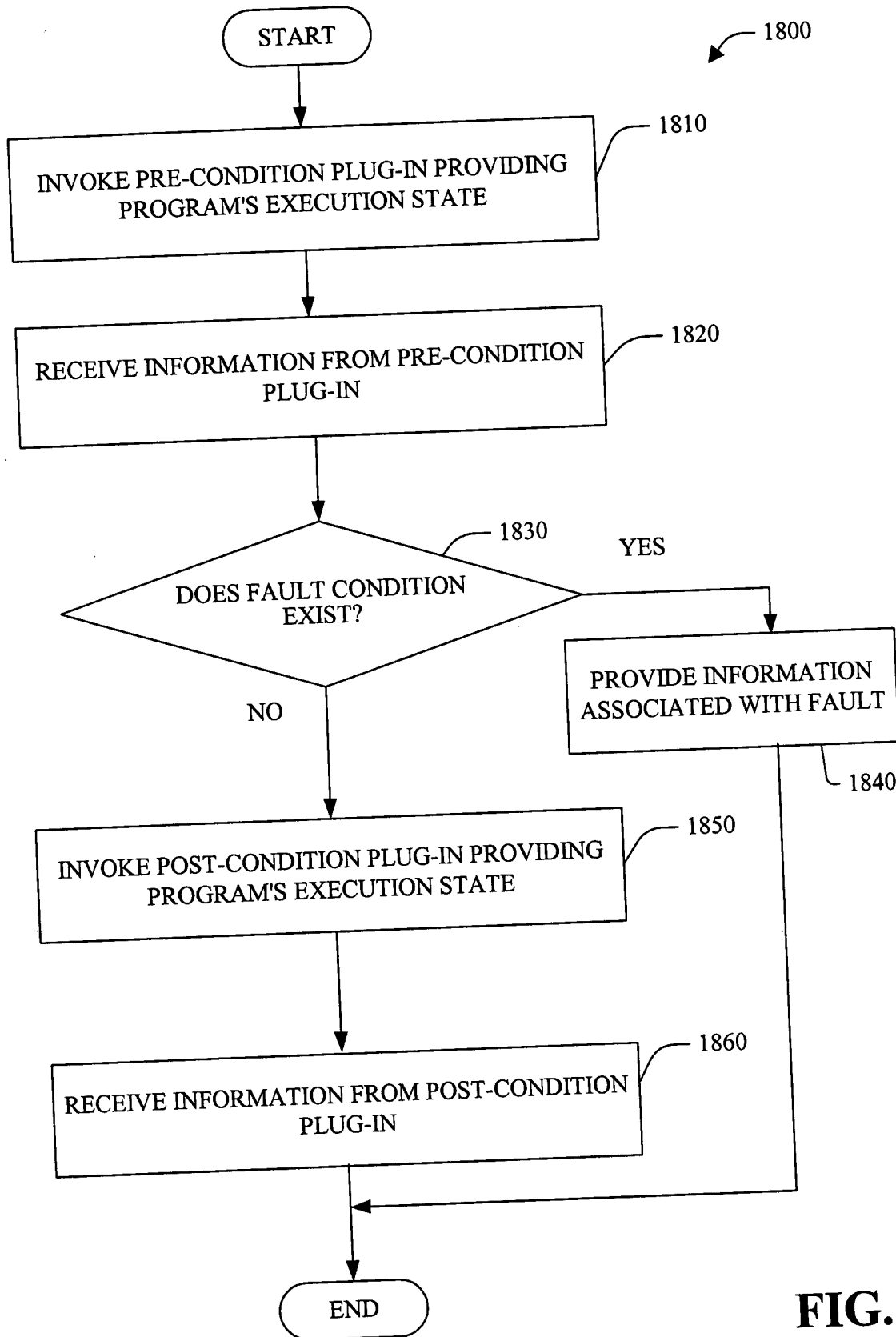


FIG. 18

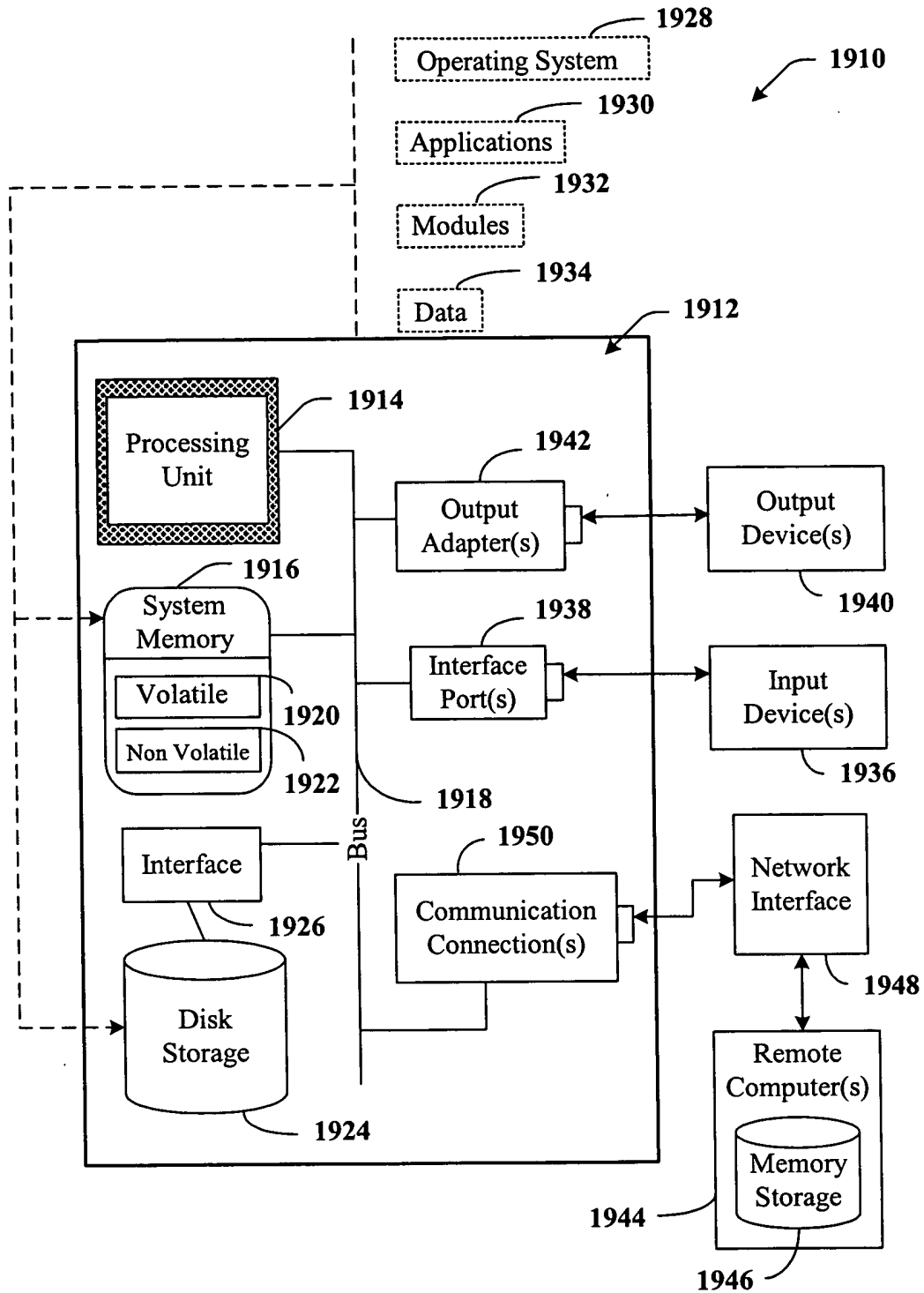


FIG. 19